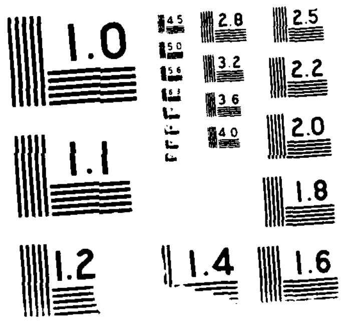
AD-A194 293 MACSYMA PROGRAM FOR THE PAINLEVE TEST OF NONLINEAR ORDINARY AND PARTIAL D (U) WISCONSIN UNIV-MADISON CENTER FOR MATHEMATICAL SCIENCES W HEREMAN ET AL 1/1

UNCLASSIFIED DEC 87 CMS-TSR-88-21 AFOSR-85-0263 F/G 12/4 NL


1.0
1.1
1.2
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

CMS Technical Summary Report #88-21

MACSYMA PROGRAM FOR THE PAINLEVÉ TEST OF NONLINEAR ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS

Willy Hereman and Eric Van den Bulck

UNIVERSITY OF WISCONSIN

CENTER FOR THE MATHEMATICAL SCIENCES

Center for the Mathematical Sciences
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53705

December 1987

(Received December 15, 1987)

Sponsored by

Air Force Office of Scientific Research
Bolling Air Force Base
Washington, DC 20332

UNIVERSITY OF WISCONSIN-MADISON
CENTER FOR THE MATHEMATICAL SCIENCES

MACSYMA PROGRAM FOR THE PAINLEVÉ TEST OF NONLINEAR ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS

Willy Hereman*,1 and Eric Van den Bulck**

Technical Summary Report #88-21
December 1987

ABSTRACT

A MACSYMA program is presented which determines whether a given single nonlinear ODE or PDE with real polynomial terms fulfills the necessary conditions for having the Painlevé property. The program listing is given, together with a synopsis of the algorithm, a model of the output of the program, a list of tested examples and instructions to use the package.

AMS (MOS) Subject Classifications: 34E15, 35G20, 68Q40

PACS Classifications: 2.30Jr, 2.90, 3.20g, 3.40k

Key Words: Painlevé property, integrability, macsyma, symbolic programming, soliton theory, dynamical systems.

*Mathematics Department, University of Wisconsin-Madison, Madison, WI 53706.

**Department of Mechanical Engineering, Solar Energy Laboratory, University of Wisconsin-Madison, Madison, WI 53706.

1This work was partially supported by the Air Force Office of Scientific Research under Grant No. 85-NM-0263.

MACSYMA PROGRAM FOR THE PAINLEVE TEST OF NONLINEAR ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS

Willy Hereman*†

and

Eric Van den Bulck‡

1. INTRODUCTION

A simple equation or system is said to have the Painlevé property, PP, if its only singularities in the complex plane consist of movable poles[1-4]. This requires that the solution f of a PDE, say in two independent variables (t,x) can be expressed as

$$f = g^{\alpha} \sum_{k=0}^{\infty} u_k g^k, \tag{1}$$

with $u_0(t,x) \neq 0$, α a negative integer, and where $u_k(t,x)$ are analytic functions in a neighbourhood of the singular, non-characteristic manifold $g(t,x) = 0$, with $g_x(t,x) \neq 0$.

Performing the Painlevé test, as proposed by Weiss[3], involves three steps:

(i) Determination of the negative integer α and u_0 from the leading order *ansatz* $f \propto u_0 g^{\alpha}$, by balancing the minimal power terms;

(ii) Indentification of the non-negative integer powers r, the *resonances*, at which arbitrary functions u_r can enter the expansion $f \propto u_0 g^{\alpha} + u_r g^{\alpha+r}$. This is achieved by requiring that u_r is arbitrary after substitution of this form for f into the equation, only retaining the most singular terms;

(iii) Verification that the correct number of arbitrary functions u_r indeed exists by substituting a truncated expansion of the form (1), $k = 1,2,...,rmax$, where $rmax$ represents the largest resonance, into the given equaton. At non-resonance

† Mathematics Department and Center for the Mathematical Sciences, University of Wisconsin-Madison, Wisconsin 53706, USA

‡ Department of Mechanical Engineering, Solar Energy Laboratory, University of Wisconsin-Madison, Wisconsin 53706, USA. Future address : Departement Werktuigkunde, Katholieke Universiteit Leuven, B-3030 Heverlee, Belgium

* Sponsored by the Air Force Office of Scientific Research under Grant No. 85-NM-0263

levels, u_k should be unambiguously determined; at resonance levels u_r should be arbitrary due to a vanishing coefficient of $g^{r+minpowg}$ (compatibility condition). Here, *minpowg* denotes the (negative) power in g of the most singular terms in the equation.

An equation or system for which the above steps can be carried out consistently, is said to have the PP and is conjectured to be integrable[1,5-7]. Counter examples[7-10] of integrable PDEs without the PP disprove the necessity of the PP for integrability (i. e. being exactly solvable by a linear integral equation of Inverse Scattering Transform); whereas some non - integrable equations seem to have the PP as defined by Weiss *et al*[3], hence questioning the conjecture[11] in its present form. Refinements of the PP have been established[12-15], allowing for rational power expansions of f, which lead to algebraic branch points in addition to movable poles.

Nevertheless, the PP has been a useful criterion for complete integrability tests[5,6,10-18]. It is a handy tool in the derivation of Bäcklund transformations and Lax pair representations for various PDEs[8,18-20], and it gives insight in Hirota's direct method for solitary wave construction[21,22], and other expansion methods which lead to rational solutions[18].

2. SCOPE AND LIMITATIONS OF THE PROGRAM

The program is largely based upon the MACSYMA routine *ODEPAINLEVE* developed by Rand and Winternitz[23], which checks the PP for a single ODE with real polynomial terms.

The package works for both single ODEs and PDEs, with arbitrary degree of nonlinearity. In principal, the number of variables is unlimited. Although the released version of the program works with only four independent variables. PDEs may have time and space dependent coefficients of integer degree. Transendental terms in the equation must be removed by a suitable exponential transformation of the dependent variable[17,24,25].

Due to 'intermediate expression swell' it may occur that the program can not perform the test in a reasonable time. Therefore, verifying the resonances and compatibility conditions are made optional through the boolian variable *do_resonances.* The highest level of verification is controllable by entering a value for *max_resonance.* For the investigation of Bäcklund transformations one may consider calculations beyond the automatically determined level *rmax* by taking *max_resonance* $\geq$ *rmax*.

Intermediate output can be obtained by including extra *print* statements in the program. (Warning : After any alternation in the main program a new LISP version needs to be created by *batch(main.p)* in MACSYMA).

Major difficulties arise when u_0 can not be substituted since it occurs in subsequent calculations at a lower degree than the one present at its evaluation. In that case, calculations by hand or in interactive MACSYMA are recommended. The vital steps for that are easily extracted from the program listing.

At present, the program does not perform the weakest forms of the Painlevé test[12-15] although α in (1) may be rational and can be positive, through a user supplied value for *beta.* Later extensions[26] will cover complex equations and coupled systems. Some of these must be treated by rather general rational power expansions[12-15]. A further refinement of the algorithm, performing selective substitutions of u_k and their derivatives in the recursive relations, will allow to construct Bäcklund transformations for PDEs[8,19,25].

3. HOW TO USE THE PROGRAM : EXAMPLES

The equation to be tested should be saved in a *batch* file, before entering MACSYMA.

Example 1 : *An ODE, where the use of independent variable x is mandatory.*

```
/* Batch file R_W_(5.6) for testing Eq. (5.6) in paper Rand-Winternitz23) */
    eq : (f^2)*fx[3](x) - 3*(fx[1](x))^3 $
    beta : -2 $
    max_resonance : 5 $
/* beta and max_resonance are specified here */
```

Note that, during the test the variable x will be replaced by $g = x - x_0$, where x_0 is the arbitrary initial value of x.

Example 2 : A PDE, with independent variables t, x, and y.

```
/* Batch file KP, for the Kadomtsev-Petviashvili Equation3) */
    eq : ftxy[1,1,0](t,x,y) + (ftxy[0,1,0](t,x,y))^2 + f*ftxy[0,2,0](t,x,y)
    + b*ftxy[0,4,0](t,x,y) + ftxy[0,0,2](t,x,y) $
/* b is a constant; beta, do resonances and max resonance are not specified */
```

Similarly, using $ftxyz[.,.,.,.](t,x,y,z)$, evolution and wave equations in t,x,y and z can be entered. For parametrical equations the explicit dependence on the variables must be given (e.g. coefficients $a(t), b(x), \ldots$). The *five first* letters of the alphabet are reserved to denote arbitrary constants, if necessary to be supplemented with $a1, a2, \ldots, b1, \ldots$.

Once the batch files are made, enter MACSYMA and subsequently type :

(c1) batch(main_p) $	→ to create *painleve.lsp*, only to be done ones
(c2) batch(setup_p) $	→ to repeat before every new example
(c3) batch(kp) $	→ reads in the batch file *kp*
(c5) writefile(test_kp) $	→ opens output file *test kp*
(c6) batch(exec_p) $	→ starts the actual Painleve test
(c7) closefile() $	→ closes output file *test_kp*

4. TEST RUNS

Although the program has been successfully tested, with eunice MACSYMA version 309.3 on a VAX 11/780, for more than 30 examples including all ODEs and selected PDEs from the cited references, unexpected situations may still occur. In case of trouble, the printout gives vital information to remedy the problems. The program does not rely on non-standard MACSYMA routines (e.g. *powers* is provided).

Model of Output :

For the Space Dependent Burgers equation[10,20] : $-x^2 f_{2x} - x f f_x + f_t = 0$, which passes the test :

```
                                          2
PAINLEVE ANALYSIS OF EQUATION,  - f      x  - f f   x + f    = 0
                                   x x           x         t

                    alfa
 SUBSTITUTE  u  g        FOR f IN ORIGINAL EQUATION.
              0
 MINIMUM POWERS OF g ARE  [2 alfa - 1, alfa - 2]

                     2 alfa - 1          2
   * COEFFICIENT OF  g             IS  - u  alfa g   x
                                          0        x
                     alfa - 2                                  2  2
   * COEFFICIENT OF  g          IS  - u  (alfa - 1) alfa (g )   x
                                       0                   x

 FOR EXPONENTS ( 2 alfa - 1 ) AND ( alfa - 2 ) OF g, DO

   WITH alfa = - 1 , POWER OF g is  - 3  --> SOLVE FOR  u
                                                         0
                                       1
      TERM  - u  g  x (2 g  x - u ) --  IS DOMINANT IN EQUATION.
               0  x       x      0   3
                                     g

   WITH  u  = 2 g  x  --> FIND RESONANCES
          0      x
                   alfa          r + alfa
   SUBSTITUTE  u  g      +  u  g          FOR f IN EQUATION
                0            r
```

TERM $-(g_x)^2 (r-2)(r+1) x^2 u_r g^{r-3}$ IS DOMINANT IN EQUATION.

THE ONLY NON-NEGATIVE INTEGRAL ROOT IS $[r = 2]$

WITH MAXIMUM RESONANCE = 2 —> CHECK RESONANCES.

SUBSTITUTE POWER SERIES $\sum_{k=0}^{2} g^{k-1} u_k$ FOR f IN EQUATION.

WITH $u_0 = 2 g_x x$

COEFFICIENT OF $\frac{1}{g^2}$ IS $2 g_x x (g_{xx} x^2 + u_1 g_x x - g_t)$

$$u_1 = - \frac{g_{xx} x^2 - g_t}{g_x x}$$

COEFFICIENT OF $\frac{1}{g}$ IS 0

u_2 IS ARBITRARY !

COMPATIBILITY CONDITION IS SATISFIED !

The results of test runs on 15 other examples are summarised in Table 1.

5. LISTING

5.1 Listing of *setup_p* routine

```
/* The batch file SETUP_P, which initializes the program after every example */
   kill (all)$
   loadprint : false$
   loadfile ("painleve.lsp")$
   derivabbrev : true$
   depends([f,eq],[t,x,y,z])$
   fx[k](x):=diff(f,x,k)$
   ftx[k,l](t,x):=diff(diff(f,t,k),x,l)$
   ftxy[k,l,m](t,x,y):=diff(diff(diff(f,t,k),x,l),y,m)$
   ftxyz[k,l,m,n](t,x,y,z):=diff(diff(diff(diff(f,t,k),x,l),y,m),z,n)$
```

5.2 Listing of *exec_p* routine

```
/* The batch file EXEC_P, to execute the test for an example */
   exec_painleve (eq, beta, do_resonances, max_resonance)$
```

5.3 Listing of *main.p* routine

```
/* The batch file MAIN_P, with the main program for the Painleve test */
define_variable (do_resonances, true, boolean)$
declare (max_resonance, integer)$

nonposint (expres) := if integerp(expres) then is(expres<=0) else false$

solveforhipow (expr,var) := block( [m],
    expr : expand(expr),
    m : hipow(expr,var),
    return(part(solve(expr,var↑m),1)))$

konstant (expr,var,q) := block ([h,co],
  co : ratcoef(expand(expr),var,0),
  if q # 1 then (
     co : ratsubst (h,var↑(1/q),co),
     co : ratcoef(co,h,0)
  ),
  return (co))$

pow(expr,var):=block (
  [power_list:'[],ff:expand(expr),tmp,tmp1,infun,inflag:true],
  declare(power_list,special),
  if not atom(ff) and part(ff,0) = "+" then infun:args(ff) else infun:[ff],
  for ii% in infun do (
    tmp1:ii%,
    if not atom(tmp1) and part(tmp1,0) = "-" then tmp1:substpart("+",tmp1,0),
    if atom(tmp1) or inpart(tmp1,0)#"*"  then tmp1:[tmp1],
    tmp:maplist(lambda([u],get_exponent(u,var)),tmp1),
    tmp:delete('[],tmp),
    if tmp = '[] then (
      if not member(0,power_list) then power_list:endcons(0,power_list)
    )
    else (
      if not member(first(tmp),power_list) then
       power_list:endcons(first(tmp),power_list)
    )
  ),
  power_list)$

get_exponent(expr,var):=block (
  if freeof(var,expr) then return('[]),
  if atom(expr) then return(1),
  if inpart(expr,0)="↑" and inpart(expr,1)=var then
    return(inpart(expr,2))
  else
    return('[]))$

check_resonances(eq,valalfa,minpowg,uzero,rmax,s) := block (
  [series,eq1,eq2,dalfa,co,uk],
  print ("    WITH MAXIMUM RESONANCE = ", rmax, " —> CHECK RESONANCES."),
  print ("    SUBSTITUTE POWER SERIES ", 'sum(u[k]*g↑(valalfa+k),k,0,rmax),
         " FOR f IN EQUATION."),
  series : sum(u[k]*g↑(valalfa+k),k,0,rmax),
  eq1 : ratsubst(series,f,eq),
  eq1 : ev(eq1,diff),
  eq1 : expand(eq1 / (g↑minpowg)),
  eq1 : ratsimp(ratsubst(0,g↑(s*rmax+1),eq1)),
  co : denom(eq1),
  if integerp(co) and co # 1 then eq1 : num(eq1),
  if uzero # [] then eq1 : ratsubst(rhs(expand(uzero)),lhs(uzero),eq1),
dalfa : denom(abs(valalfa)),
co : konstant (eq1,g,dalfa),
if co # 0 then (
   print("*** FATAL ERROR : NON-ZERO LEADING COEFFICIENT REMAINS ",
         "AFTER SUBSTITUTING ", uzero, ". ***"),
   return ()
),
if dalfa # 1 then eq1 : radcan(eq1),
if uzero # [] then print("    WITH", uzero)
              else print("    WITH", u[0], "ARBITRARY !"),
for i : 1 thru s*rmax do block ([k],
  if dalfa # 1 then for j : 1 thru dalfa-1 do (
       co : ratcoef(expand(eq1),g,j/dalfa),
       if  co # 0 then (
          if not freeof(g,co) then (
             print("*** FATAL ERROR : COEFFICIENT STILL CONTAINS g. ***"),
             return
          ). /* end if */
          eq1 : radcan(eq1 - co*g↑(j/dalfa))
  )     ),
  eq1 : ratsimp(eq1/g),
  co : konstant (eq1,g,dalfa),
  eq1 : eq1 - co,
  co : ratsimp(xthru(co)),
  if not integerp(co) then co : factor(co),
  print("        COEFFICIENT OF",g↑(i+minpowg),"IS ",co),
  k : i/s,
  if co # 0 and integerp(k) then (
     if ratcoef(co,u[k],s) = 0 then (
        if not freeof(u[k],co) then
           print("*** RESONANCE VIOLATED BY APPEARANCE OF",u[k],
                 "FROM NON-LEADING TERMS. ***")
        else block( [top,j,cox0],
           print("        ", u[k]↑s, " IS ARBITRARY !"),
           print("        COMPATIBILITY CONDITION: ", co, " = 0"),
           if (ratsimp(ev(co, diff)) = 0) then
              print("        CONDITION IS SATISFIED !")
           else (
              print("   *** CONDITION NOT SATISFIED. ***"),
              print("   *** CHECK FOR FREE PARAMETERS OR PRESENCE OF",
                    u[0],". ***")),
           top : num(co),
           if not freeof(x0,top) then for j:0 while top # 0 do (
              top : expand(top),
              cox0 : coeff(top,x0,0),
              top : (top-cox0)/x0,
              cox0 : factor(cox0),
              print("        COEFFICIENT OF ",x0↑j,"  IN NUMERATOR IS",cox0)
     )  )   ) /* end do */
     else block ([temp],
        uk: solve(co,u[k]↑s),
        uk : part(uk,1),
        temp : factor(ev(rhs(uk),diff)),
        uk : ratsubst(temp,rhs(uk),uk),
        print("        ",uk),
        eq1 : ratsubst(rhs(uk),lhs(uk),eq1),
        if uzero # [] then eq1 : ratsubst(rhs(expand(uzero)),lhs(uzero),eq1),
        eq1 : ev(eq1,diff)
  )  ) /* end else if */
  else if co = 0 and integerp(k) then (
       print ("       ", u[k]ts, "IS ARBITRARY !"),
       print ("       COMPATIBILITY CONDITION IS SATISFIED !")
    ) ))$ /* end of function check_resonances */

  find_resonances(eq,valalfa,minpowg,uzero) := block (
    [eq1,eq2,coefr,s,i,ii,temp1,temp2,powU0,dalfa],
    print ("    WITH ",uzero, " --> FIND RESONANCES"),
    print ("    SUBSTITUTE ", u[0]*gtalfa, " + ", u[r],gt(alfa+r),
         " FOR f IN EQUATION"),
    eq1 : ratsubst(u[0]*(gtvalalfa)+u[r]*(gt(r+valalfa)),f,eq),
    eq1 : ev(eq1,diff),
    eq1 : ratsimp(eq1/(gtminpowg)),
    dalfa : denom (abs(valalfa)),
    if dalfa # 1 then eq1 : radcan(eq1),
    powU0 : hipow(lhs(uzero),u[0]),
    temp1 : denom(eq1), if integerp(temp1) and temp1 # 1 then eq1 : num(eq1),
    temp2 : hipow(subst(r=1,eq1),g),
    coefr : 0,
    for i : 1 thru temp2 while coefr = 0 do (
      eq2 : ratcoef(eq1,u[r],i),
      eq2 : ratcoef(eq2,g,i*r),
      coefr : konstant (eq2,g,dalfa),
      if uzero # [] then (
         coefr : expand(coefr),
         while hipow(coefr,u[0]) >= powU0 do
            coefr : expand(ratsubst(rhs(expand(uzero)), lhs(uzero), coefr))
      ),
      s : i
    ),
    coefr : factor(num(ratsimp(radcan(coefr/u[r]ts)))),
    print ("       TERM ",coefr,u[r]ts,gt(s*r+minpowg)," IS DOMINANT IN EQUATION."),
    coefr:xthru(coefr/factor(gcd(coefr,diff(coefr,r),r))),
    if part(coefr,0) = "-" then coefr : - coefr,
    if part(coefr,0) = "/" then coefr : num(coefr),
    temp2 : 1,
    if part(coefr,0) = "+" then (
      if hipow(coefr,r) = 1 then (
        if (coeff(coefr,r) = 1 and nonposint(subst(r=0,coefr))) then
           temp2 : coefr
    ) )
    else if part(coefr,0) = "*" then
      for ii : 1 thru length(coefr) do block( [pp],
        pp : part(coefr,ii),
        if hipow(pp,r) = 1 then (
          if coeff(pp,r) = 1 and nonposint(subst(r=0,pp)) then
            temp2 : temp2 * pp
    )   ),
    if temp2 = 1 then
      print ("       THERE ARE NO NON-NEGATIVE INTEGRAL ROOTS FOR r.")
    else (
      temp2 : solve(temp2,r),
      if length(temp2) = 1 then
        print ("       THE ONLY NON-NEGATIVE INTEGRAL ROOT IS ", temp2)
      else
        print ("       THE",length(temp2),"NON-NEGATIVE INTEGRAL ROOTS ARE ", temp2),
      print (" "),
      temp2 : apply(max,map(rhs,temp2)),
      if temp2 > 0 then (
        if do_resonances then (
           if integerp(max_resonance) then temp2 : max_resonance,
          check_resonances(eq,valalfa,minpowg,uzero,temp2,s)
       )
       else
          print ("    EXIT FIND_RESONANCES WITHOUT CHECKING.")
   ) ))$ /* end of function find_resonances */

find_uzero(eq,valalfa,minpowg) := block( [eq2,r,lowest,uzero,dalfa],
  print ("    WITH alfa = ", valalfa, ", POWER OF g is ", minpowg,
         " —> SOLVE FOR ", u[0]),
  eq2 : ratsubst(u[0]*(g↑valalfa),f,eq),
  eq2 : ev(eq2,diff),
  eq2 : ratsimp(eq2 / (g↑minpowg)),
  dalfa : denom (abs(valalfa)),
  if dalfa # 1 then eq2 : radcan(eq2),
  lowest : denom(eq2),
  if integerp(lowest) and lowest # 1 then  eq2 : num(eq2),
  lowest : factor(konstant (eq2,g,dalfa)),
  print ("        TERM ",lowest, g↑minpowg, " IS DOMINANT IN EQUATION."),
  if lowest = 0 then  find_resonances (eq,valalfa,minpowg,[])
  else if freeof(u[0],lowest) then
         print("***  FAILURE OF PAINLEVE TEST — NONZERO LEADINGORDER IS ",
               "FREE OF",u[0],", ABANDON THIS alfa VALUE. ***")
  else (
    for i:1 while remainder(lowest,u[0]) = 0 do lowest : xthru(lowest/u[0]),
    if freeof(u[0],lowest) then
       print("  ***  FAILURE OF PAINLEVE TEST, ", u[0]=0,
             ", ABANDON THIS alfa VALUE. ***")
    else (
        if (denom(lowest) # 1) then lowest : num(lowest),
        lowest : factor(lowest/(gcd(lowest,diff(lowest,u[0]),u[0]))),
        if part(lowest,0) = "-" then lowest : -lowest,
        if part(lowest,0) = "/" then lowest : num(lowest),
        if part(lowest,0) # "*" then (
           uzero : solveforhipow(lowest,u[0]),
           uzero : ratsubst(factor(rhs(uzero)),rhs(uzero),uzero),
           find_resonances(eq,valalfa,minpowg,uzero)
         )
         else for i:1 thru length(lowest) do (
                  uzero : part(lowest,i),
                  if not freeof(u[0],uzero) then (
                     uzero : solveforhipow(uzero,u[0]),
                     find_resonances(eq,valalfa,minpowg,uzero)
    ) )  )          ))$ /* end of function find_uzero */

painleve (eq, beta, do_resonances, max_resonance) := block (
  [eq1, u0, alfa, i, exp, list, len, eq2, alfalist],
  print ("————————————————————————————————————————————————————————————————"),
  print ("PAINLEVE ANALYSIS OF EQUATION, ", eq, " = 0"),
  print ("————————————————————————————————————————————————————————————————"),
  print ("  SUBSTITUTE ", u[0]*g↑alfa, " FOR f IN ORIGINAL EQUATION." ),
  eq1 : ratsubst(u[0]*g↑alfa,f,eq),
  eq1 : ev(eq1,diff),
  eq1 : map(factor,eq1),
  i : hipow(expand(denom(eq1)),g),
  list : pow(num(eq1),g),
  list : map(lambda([entry],entry-i),list),
  list : apply(min,list),
  if part(list,0) = min then list : args(list) else list : [list],
  len : length(list),
  print("  MINIMUM POWERS OF g ARE ",list),
   exp : expand(eq1),
   for  i:1 thru len do (
     eq2 : expand(ratcoef(exp,g,part(list,i))),
     eq2 : ratcoef(eq2,g,0),
     eq2 : ratsubst(0,g^alfa,eq2),
     eq2 : factor(eq2),
     print("    * COEFFICIENT OF ",g^part(list,i), " IS ", eq2)
   ),
   alfalist:[],
   if len > 1 and not ratnump (beta) then
    for i : 1 thru len-1 do for j:i+1 thru len do
     block( [valalfa,list1,minpowg],
     print ("  ------------------------------------------------------------------"),
     valalfa : solve(part(list,i)=part(list,j),alfa),
     valalfa : rhs(part(valalfa,1)),
     if valalfa < 0 then (
        list1 : subst(alfa=valalfa,list),
        minpowg : part(list1,i),
        if not member(valalfa,alfalist) then (
          print ("  FOR EXPONENTS (",part(list,i),") AND (",part(list,j),") OF g, DO"),
          alfalist : cons(valalfa,alfalist),
          if apply(min,list1) # minpowg then
            print("    POWER OF g IS NOT MINIMAL -- SKIP THIS VALUE OF ALFA")
          else  find_uzero(eq,valalfa,minpowg)
        )
        else
          print ("  FOR EXPONENTS (",part(list,i),") AND (",part(list,j),") OF g,",
                 "  alfa = ", valalfa, " ALREADY DONE!")
     )
     else print("  FOR EXPONENTS (",part(list,i),") AND (",part(list,j),
                ") OF g,  alfa = ",valalfa," IS NON-NEGATIVE.")

   ),  /* This is the end of the first line including the block */
   print ("  ------------------------------------------------------------------"),
   list : subst(beta,alfa,list),
   list : apply(min,list),
   if ratnump(beta) then (
      print("  RESULTS FOR USER-SUPPLIED VALUE alfa =",beta),
      find_uzero(eq,beta,list)
   )
)$ /* end of function painleve */

exec_painleve (eq, beta, do_resonances, max_resonance) := block (
    [degpdft,degpdfx,degpdfy,degpdfz],
    degpdft : derivdegree(eq,f,t),
    degpdfx : derivdegree(eq,f,x),
    degpdfy : derivdegree(eq,f,y),
    degpdfz : derivdegree(eq,f,z),
    if (degpdft=0) and (degpdfy=0) and (degpdfz=0) then block ([i,degx,eq1],
       remove([f,g,eq],dependency),
       depends(f,g),
       depends(g,x),
       depends(u,k),
       declare(g,mainvar),
       eq1 : ev(eq,diff),
       eq1 : ratsubst(1,diff(g,x),eq1),
       degx :  derivdegree(eq1,g,x),
       for i:2 thru degx do (
         eq1 : ratsubst(0,diff(g,x,ev(i)),eq1),
         eq1 : num(xthru(eq1))
      ),
      eq : subst(g+x0,x,eq1),
      print ("  SUBSTITUTE ", x, "-->", g+x0)
    )
    else (
      depends ([g],[t,x,y,z]),
      depends([u],[k,t,x,y,z])
    ),
    painleve (eq, beta, do_resonances, max_resonance))$

  save (["painleve.lsp"], do_resonances, functions);
```

ACKNOWLEDGEMENTS

This work is partially supported by AFOSR under Grant 85-NM-0263. We thank Jonathan Len (Symbolics Inc.) for writing the POWERS subroutine.

ODE or PDE	REF	ALPHA	u_0	$r \geq 0$	u_i $i \geq 1$	COMP	PP
Korteweg-deVries $f_t+ff_x+bf_{3x}=0$	3	-2	$u_0=-12bg_x^2$	4, 6	u_1, u_3, u_5 det u_4, u_6 arb	satis	yes
sine-Gordon $2ff_{tx}-2f_tf_x-f^3+f=0$	27	-2	$u_0=4g_tg_x$	2	u_1 det u_2 arb	satis	yes
Kadomtsev-Petviashvili or two-dimensional KdV $f_{tx}+f_x^2+ff_{2x}+bf_{4x}+f_{2y}=0$	3	-2	$u_0=-12bg_x^2$	4, 5, 6	u_1, u_2, u_3 det u_4, u_5, u_6 arb	satis	yes
Harry Dym $f_t-f^3f_{3x}=0$	8	$\frac{2}{3}$	$u_0^3=\frac{9g_t}{4g_x^3}$	$\frac{2}{3}, \frac{4}{3}$	—	—	no
Bullough-Dodd $ff_{xt}-f_tf_x+af-bf^4=0$	19	-1	$u_0^2=\frac{g_xg_t}{b}$	2	u_1 det u_2 arb	par dep	und
$f_t+bf_xf_{2x}+f_{5x}=0$	28	-2	$u_0=-\frac{60g_x^2}{b}$	2, 6, $r^2-13r+60$	u_1, u_3, u_4, u_5 det u_2, u_6 arb	satis	und
Parametrical sinh-Gordon $ff_{tx}-f_tf_x-\frac{b(t)}{2}(f^3-f)$ $+a(t)(ff_x+xff_{2x}-xf_x^2)=0$	15	-2	$u_0=\frac{4g_x}{b(t)}\cdot(g_t+a(t)xg_x)$	2	u_1 det u_2 arb	satis	yes

Table 1 Test Cases

ODE or PDE	REF	ALPHA	u_0	$r \geq 0$	u_i $i \geq 1$	COMP	PP
$f_{2x} + \frac{af_x}{x} + bf^3 = 0$	23	-1	$u_0^2 = -2/b$	4	u_1, u_2, u_3 det u_4 arb	par dep	und
$(f_{2x} + aff_x)^2 + (cx+b)f_x^2$ $+ (ex+d)f = 0$	23	-1 $\frac{2}{3}, \frac{3}{2}, 4$	$u_0 = 2/a$	2	u_1^2 det u_2 arb	satis	yes
$f^2 f_{3x} - 3f_x^3 = 0$	23	$\beta = -2$	u_0 arb	0, 10	$u_i = 0$ $1 \leq i \leq 9$ u_{10} arb	satis	yes
Emden type $f_{2x} + 3ff_x + f^3 = 0$	29	-1	$u_0 = 1$ $u_0 = 2$	1 none	u_1 arb —	satis —	yes no
Ince type $f_{2x} + ff_x - f = 0$	5	-1	$u_0 = 2$	2	$u_1 = 0$ u_2 arb	not satis	no
Painleve II $f_{xx} - 2f^3 - xf - a = 0$	30	-1	$u_0 = \pm 1$	4	u_1, u_2, u_3 det u_4 arb	satis	yes
Double sine-Gordon type $ff_{2x} - f_x^2 - a(f^3 - f) + 1 - f^4 = 0$	27	-1 1	$u_0 = \pm 1$ $u_0 = \pm 1$	2 2	u_1 det, u_2 arb u_1 det, u_2 arb	satis satis	yes yes
Diffusion $f_t - (f^{-2} f_x)_x = 0$	16	$-\frac{1}{2}$	$u_0 = -g_x^2/(2g_t)$	$\frac{1}{2}$	—	—	no

<u>Table 1</u> cont.

6. REFERENCES

1) Ablowitz, M.J., Ramani, A. and Segur, H., *Lett. Nuovo Cim.* **23**, 333 (1978).
2) Ablowitz, M.J., Ramani, A. and Segur, H., *J. Math. Phys.* **21**, 715 (1980).
3) Weiss, J., Tabor, M. and Carnevale, G., *J. Math. Phys.* **24**, 522 (1983).
4) Ward, R.S., *Phys. Lett.* **102A**, 279 (1984).
5) McLeod, J.B. and Olver, P.J., *Siam J. Math. Anal.* **14**, 488 (1983).
6) Menyuk, C.R., Chen, H.H. and Lee, Y.C., *Phys. Rev. A* **27**, 1597 (1983).
7) Lakshmanan, M. and Kaliappan, P., *J. Math. Phys.* **24**, 795 (1983).
8) Weiss, J., *J. Math. Phys.* **24**, 1405 (1983).
9) Steeb, W.-H., Kloke, M., Spieker, B.M. and Grensing, D., *Prog. Theor. Phys.* **73**, 344 (1985).
10) Steeb, W.-H., and Louw, J.A., *Physica Scripta* **36**, 11 (1987).
11) Doktorov, E.V. and Sakovich, S. Yu., *J. Phys. A* **18**, 3327 (1985).
12) Dorizzi, B., Grammaticos, B. and Ramani, A., *J. Math. Phys.* **24**, 2282 (1983).
13) Grammaticos, B., Dorizzi, B. and Ramani, A., *J. Math. Phys.* **24**, 2289 (1983).
14) Graham, R., Roekaerts., D. and Tél, T., *Phys. Rev. A***31**, 3364 (1985).
15) Chowdhury, Roy A., Chanda, P.K. and Roy, S., *Prog. Theor. Phys.* **75**, 751 (1986).
16) Steeb, W.-H., Grauel, A., Kloke, M. and Spieker, B.M., *Physica Scripta* **31**, 5 (1985).
17) Steeb, W.-H., Kloke, M., Spieker, B.M. and Grensing, D., *Z. Phys. C* **28**, 241 (1985).
18) Weiss, J., *J. Math. Phys.* **25**, 13 (1984).
19) Weiss, J., *J. Math. Phys.* **27**, 1293 (1986).
20) Steeb, W.-H., Louw, J.A. and Strampp, W., *Prog. Theor. Phys. Lett.* **75**, 455 (1986).
21) Gibbon, J.D., Radmore, P., Tabor, M. and Wood, D., *Stud. Appl. Math.* **72**, 39 (1985).
22) Steeb, W.-H., Kloke, M. and Spieker, B.M., *J. Phys. A* **17**, L825 (1984).
23) Rand, D.W. and Winternitz, P., *Comp. Phys. Comm.* **42**, 359 (1986).
24) Steeb, W.-H., Kloke, M., Spieker, B.M. and Grauel, A., *Lett. Nuovo Cim.* **39**, 429 (1984).
25) Steeb, W.-H., Kloke, M. and Spieker, B.M., *Z. Naturforsch.* **38a**, 1054 (1983).
26) Hereman, W. and Van Den Bulck, E., *Comp. Phys. Comm.*, in preparation (1988).
27) Weiss, J., *J. Math. Phys.* **25**, 2226 (1984).
28) Hereman, W., Banerjee, P.P., Korpel, A., Assanto, G., Van Immerzeele, A. and Meerpoel, A., *J. Phys. A* **19**, 607 (1986).
29) Mahomed, F.M., *Ph.D. Thesis, University of the Witwatersrand, Johannesburg, South Africa*, App. C, 88 (1986).
30) Hlavatý, L., *Comp. Phys. Comm.* **42**, 427 (1986).